EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

As demonstrated in the chart below, ASUS directly and indirectly infringes at least claim 1 of US 9,036,701 (the "'701 Patent"). ASUS directly infringes, contributes to the infringement of, and/or induces infringement of the '701 Patent by making, using, selling, offering for sale, and/or importing into the United States the Accused Products that are covered by one or more claims of the '701 Patent. The Accused Products are devices that decode H.265-compliant video. For example, the ASUS Q543MV Notebook ("ASUS Q543MV") is a representative product for other ASUS devices that decode H.265-compliant video.

The ASUS Q543MV contains at least one video decoder that helps decode H.265-compliant video.[1] While evidence from the ASUS Q543MV is specifically charted herein, the evidence and contentions charted herein apply equally to the other ASUS Accused Products that decode H.265-compliant video.

No part of this exemplary chart construes, or is intended to construe, the specification, file history, or claims of the '701 Patent. Moreover, this exemplary chart does not limit, and is not intended to limit, Nokia's infringement positions or contentions.

The following infringement chart includes exemplary citations to ITU-T Rec. H.265 (12/2016) High efficiency video coding (available at https://www.itu.int/rec/T-REC-H.265-201612-S/en) (the "H.265 Standard"). The cited functionality has been included in editions of the H.265 Standard since April 2013 and remains in current editions of the H.265 Standard. Any ASUS device that includes a decoder that practices the functionality in any of these editions of the H.265 Standard ("H.265 Decoder") practices the claims of the '701 Patent. Thus, the Accused Products each practice the H.265 Standard and are covered by claims of the '701 Patent.

Nokia contends each of the following limitations is met literally, and, to the extent a limitation is not met literally, it is met under the doctrine of equivalents.[2]

[1] See, e.g., https://www.asus.com/us/laptops/for-home/everyday-use/asus-vivobook-pro-15-oled-q543/techspec/; https://www.intel.com/content/www/us/en/products/sku/236849/intel-core-ultra-9-processor-185h-24m-cache-up-to-5-10-ghz/specifications.html; https://developer.nvidia.com/video-encode-and-decode-gpu-support-matrix-new.

[2] This claim chart is based on the information currently available to Nokia and is intended to be exemplary in nature. Nokia reserves all rights to update and elaborate its infringement positions, including as Nokia obtains additional information during the course of discovery.

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

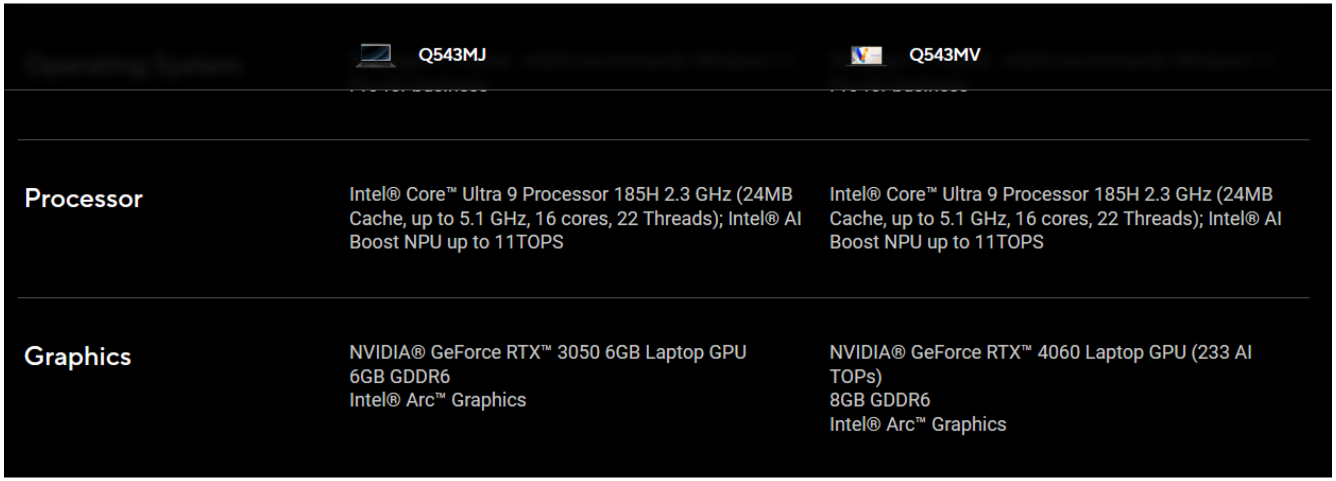
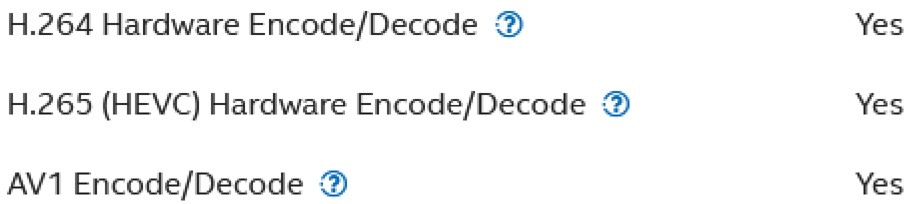
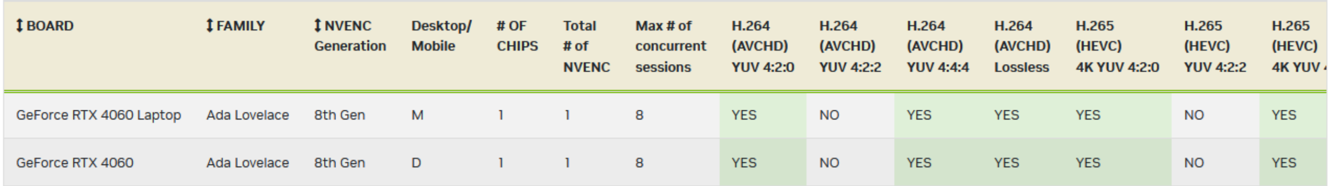
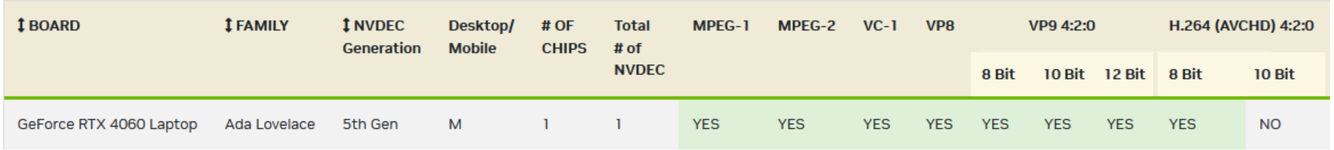
U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
1. [Pre] A method comprising:	Each of the Accused Products, such as the ASUS Q543MV, performs a method comprising the limitations below. For example, and without limitation, the Asus Q543MV uses hardware-accelerated decoding and includes an NVIDIA GeForce RTX 4060 Laptop graphics processing unit ("GPU") and an Intel Core Ultra 9 Processor 185H.  Q543MJ | Q543MV Processor | Intel® Core™ Ultra 9 Processor 185H 2.3 GHz (24MB Cache, up to 5.1 GHz, 16 cores, 22 Threads); Intel® AI Boost NPU up to 11TOPS | Intel® Core™ Ultra 9 Processor 185H 2.3 GHz (24MB Cache, up to 5.1 GHz, 16 cores, 22 Threads); Intel® AI Boost NPU up to 11TOPS Graphics | NVIDIA® GeForce RTX™ 3050 6GB Laptop GPU 6GB GDDR6 Intel® Arc™ Graphics | NVIDIA® GeForce RTX™ 4060 Laptop GPU (233 AI TOPs) 8GB GDDR6 Intel® Arc™ Graphics Source: https://www.asus.com/us/laptops/for-home/everyday-use/asus-vivobook-pro-15-oled-q543/techspec/ (last accessed March 6, 2025).  H.264 Hardware Encode/Decode ? | Yes H.265 (HEVC) Hardware Encode/Decode ? | Yes AV1 Encode/Decode ? | Yes

Exhibit 15
United States Patent No. 9,036,701
Claim Chart for Infringement of Claim 1 by ASUS Accused Products

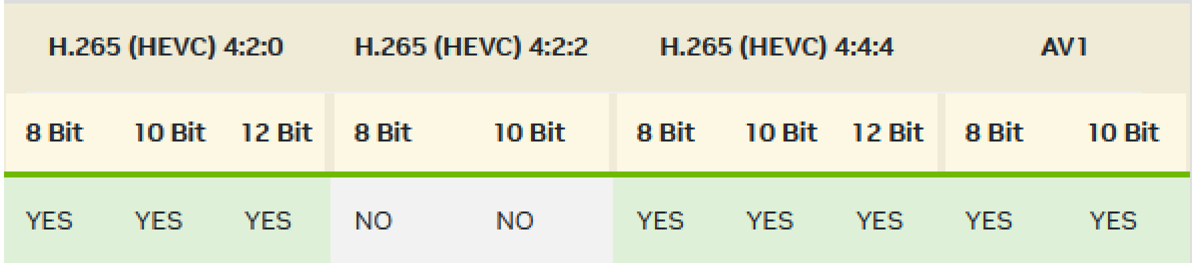
U.S. Patent No. 9,036,701	ASUS Accused Products
	Source: https://www.intel.com/content/www/us/en/products/sku/236849/intel-core-ultra-9-processor-185h-24m-cache-up-to-5-10-ghz/specifications.html (last accessed March 6, 2025) (specifications for Intel Core Ultra 9 185H).



↕ BOARD	↕ FAMILY	↕ NVENC Generation	Desktop/ Mobile	# OF CHIPS	Total # of NVENC	Max # of concurrent sessions	H.264 (AVCHD) YUV 4:2:0	H.264 (AVCHD) YUV 4:2:2	H.264 (AVCHD) YUV 4:4:4	H.264 (AVCHD) Lossless	H.265 (HEVC) 4K YUV 4:2:0	H.265 (HEVC) YUV 4:2:2	H.265 (HEVC) 4K YUV
GeForce RTX 4060 Laptop	Ada Lovelace	8th Gen	M	1	1	8	YES	NO	YES	YES	YES	NO	YES
GeForce RTX 4060	Ada Lovelace	8th Gen	D	1	1	8	YES	NO	YES	YES	YES	NO	YES



↕ BOARD	↕ FAMILY	↕ NVDEC Generation	Desktop/ Mobile	# OF CHIPS	Total # of NVDEC	MPEG-1	MPEG-2	VC-1	VP8	VP9 4:2:0			H.264 (AVCHD) 4:2:0	
										8 Bit	10 Bit	12 Bit	8 Bit	10 Bit
GeForce RTX 4060 Laptop	Ada Lovelace	5th Gen	M	1	1	YES	YES	YES	YES	YES	YES	YES	YES	NO



H.265 (HEVC) 4:2:0			H.265 (HEVC) 4:2:2		H.265 (HEVC) 4:4:4			AV1	
8 Bit	10 Bit	12 Bit	8 Bit	10 Bit	8 Bit	10 Bit	12 Bit	8 Bit	10 Bit
YES	YES	YES	NO	NO	YES	YES	YES	YES	YES

Source: https://developer.nvidia.com/video-encode-and-decode-gpu-support-matrix-new (last accessed March 6, 2025) (row for 4060 Laptop GPU).

For example, an ASUS Q543MV was used to playback an H.265-compliant video.

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

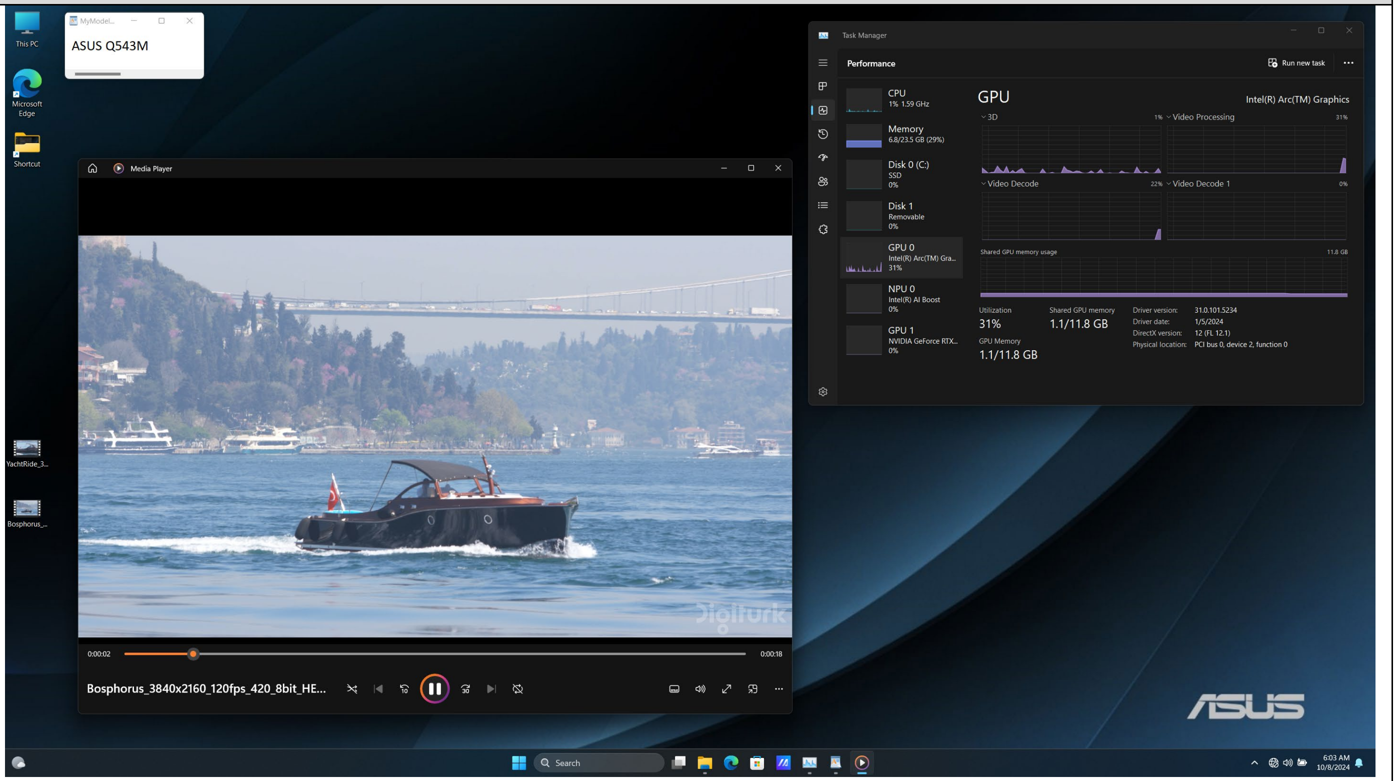
U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
	 This PC / Microsoft Edge / Shortcut / MyModel... / ASUS Q543M / YachtRide_3... / Bosphorus_... / Media Player / 0:00:02 / 0:00:18 / Bosphorus_3840x2160_120fps_420_8bit_HE... / Digiturk / ASUS / Search / 6:03 AM / 10/8/2024 / Task Manager / Performance / Run new task / 3D 1% / Video Processing 31% / Video Decode 22% / Video Decode 1 0% / Shared GPU memory usage 11.8 GB CPU 1% 1.59 GHz Memory 6.8/23.5 GB (29%) Disk 0 (C:) SSD 0% Disk 1 Removable 0% GPU 0 Intel(R) Arc(TM) Gra... 31% NPU 0 Intel(R) AI Boost 0% GPU 1 NVIDIA GeForce RTX... 0% GPU Intel(R) Arc(TM) Graphics Utilization 31% Shared GPU memory 1.1/11.8 GB GPU Memory 1.1/11.8 GB Driver version: 31.0.101.5234 Driver date: 1/5/2024 DirectX version: 12 (FL 12.1) Physical location: PCI bus 0, device 2, function 0 Source: Screenshot of an ASUS Q543MV playing back an H.265-compliant video. For example, and without limitation, the H.265 Standard specifies the following regarding the decoding process. Each of the Accused Products performs a method comprising the limitations below. **3 Definitions** For the purposes of this Recommendation | International Standard, the following definitions apply.

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
	. . . **3.12 bitstream**: A sequence of bits, **. . .** , that forms the representation of *coded pictures* and associated data forming one or more coded video sequences *(CVSs)*. . . . **3.25 coded picture**: A *coded representation* of a *picture* **. . .** . . . **3.44 decoding process**: The process specified in this Specification that reads a *bitstream* and derives *decoded pictures* from it. ITU-T Rec. H.265 (12/2016) High efficiency video coding at pp. 4 – 7.
[a] determining a frequency of occurrence threshold based on an expected frequency of occurrence of syntax elements in a bit stream;	Each of the Accused Products, such as the ASUS Q543MV, performs a method comprising determining a frequency of occurrence threshold based on an expected frequency of occurrence of syntax elements in a bit stream. For example, and without limitation, the H.265 Standard specifies the following regarding the decoding process. Each of the Accused Products performs a method comprising determining a frequency of occurrence threshold based on an expected frequency of occurrence of syntax elements in a bit stream. The following specifications provide further evidence of how each of the Accused Products operates: **3 Definitions** For the purposes of this Recommendation | International Standard, the following definitions apply. . . .

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

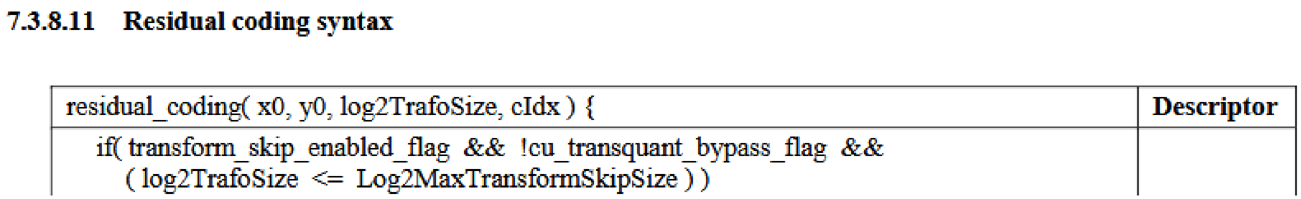
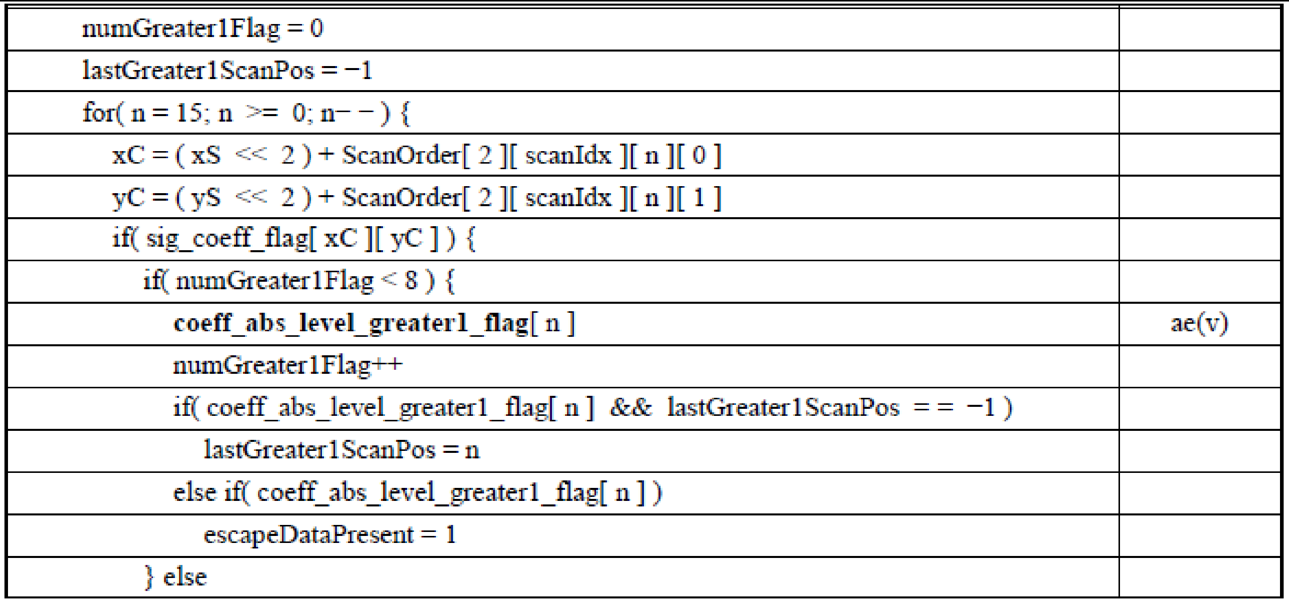
U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
	3.153 syntax element*: An element of data represented in the bitstream.* … ***3.166 transform coefficient:*** A Scalar quantity, considered to be in a frequency domain, that is associated with a particular one-dimensional or two-dimensional *frequency index* in a *transform* in the *decoding process*. ***3.167 transform coefficient level:*** *An integer quantity representing the value associated with a particular two-dimensional frequency index in the decoding process prior to scaling for computation of a transform coefficient value.* ITU-T Rec. H.265 (12/2016) High efficiency video coding at p. 12.  **7.3.8.11 Residual coding syntax** residual_coding(x0, y0, log2TrafoSize, cIdx) { — **Descriptor** if(transform_skip_enabled_flag && !cu_transquant_bypass_flag && (log2TrafoSize <= Log2MaxTransformSkipSize)) . . .

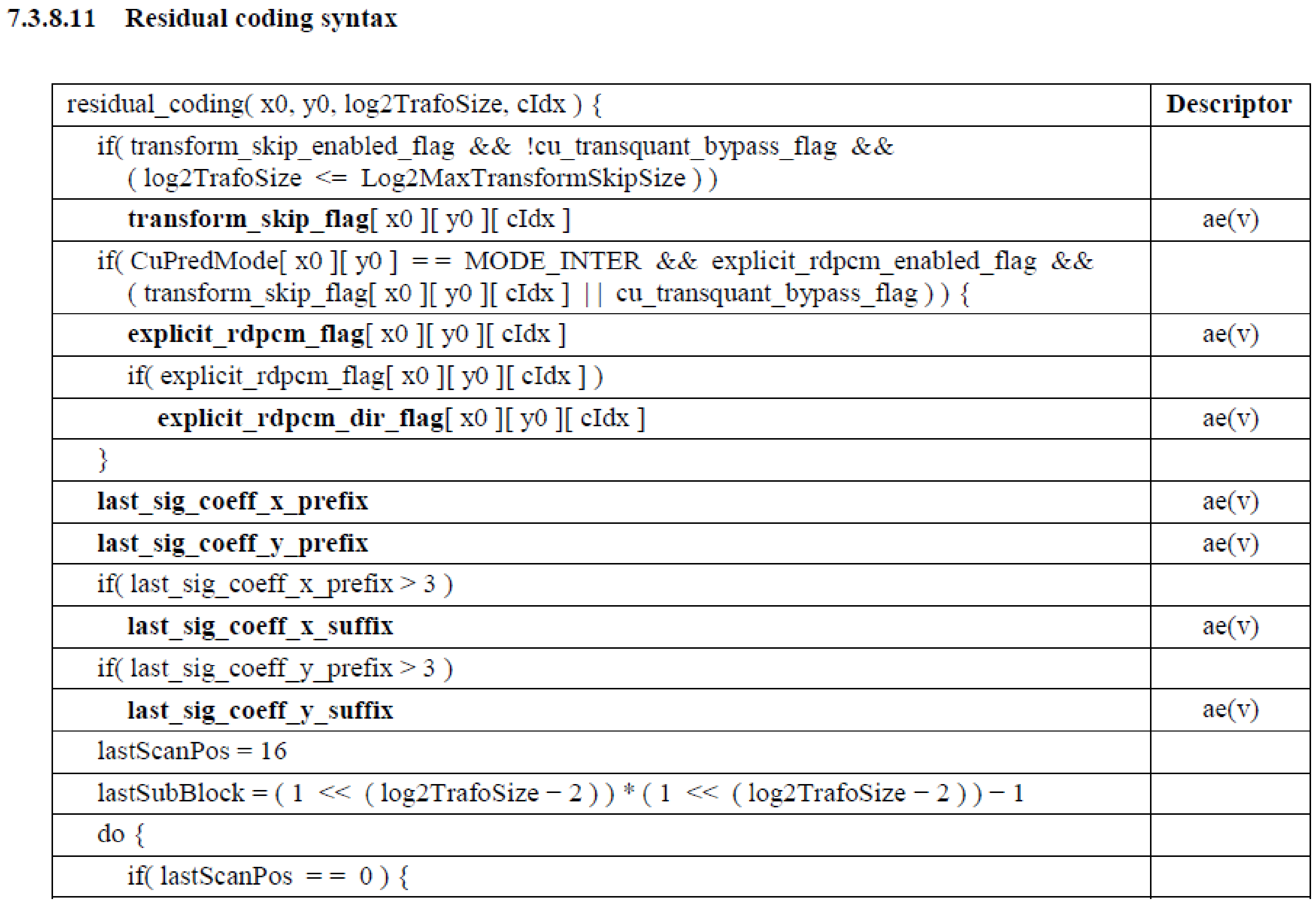
EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

<table>
<tr><th>U.S. PATENT NO. 9,036,701</th><th>ASUS ACCUSED PRODUCTS</th></tr>
<tr><td></td><td>

<table>
<tr><td>numGreater1Flag = 0</td><td></td></tr>
<tr><td>lastGreater1ScanPos = −1</td><td></td></tr>
<tr><td>for(n = 15; n &gt;= 0; n− −) {</td><td></td></tr>
<tr><td>xC = (xS &lt;&lt; 2) + ScanOrder[2][scanIdx][n][0]</td><td></td></tr>
<tr><td>yC = (yS &lt;&lt; 2) + ScanOrder[2][scanIdx][n][1]</td><td></td></tr>
<tr><td>if(sig_coeff_flag[xC][yC]) {</td><td></td></tr>
<tr><td>if(numGreater1Flag &lt; 8) {</td><td></td></tr>
<tr><td><b>coeff_abs_level_greater1_flag</b>[n]</td><td>ae(v)</td></tr>
<tr><td>numGreater1Flag++</td><td></td></tr>
<tr><td>if(coeff_abs_level_greater1_flag[n] &amp;&amp; lastGreater1ScanPos = = −1)</td><td></td></tr>
<tr><td>lastGreater1ScanPos = n</td><td></td></tr>
<tr><td>else if(coeff_abs_level_greater1_flag[n])</td><td></td></tr>
<tr><td>escapeDataPresent = 1</td><td></td></tr>
<tr><td>} else</td><td></td></tr>
</table>
[…]
ITU-T Rec. H.265 (12/2016) High efficiency video coding at pp. 58-59.
</td></tr>
<tr><td><b>[b]</b> categorizing a plurality of syntax elements of video content into first and second categories based on the frequency of occurrence threshold, wherein syntax elements which occur greater than the frequency of</td><td>Each of the Accused Products, such as the ASUS Q543MV, performs a method comprising categorizing a plurality of syntax elements of video content into first and second categories based on the frequency of occurrence threshold, wherein syntax elements which occur greater than the frequency of occurrence threshold are categorized into the second category and syntax elements which occur less than the frequency of occurrence are categorized into the first category.

For example, and without limitation, the H.265 Standard specifies the following regarding the decoding process. Each of the Accused Products performs a method comprising</td></tr>
</table>

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
occurrence threshold are categorized into the second category and syntax elements which occur less than the frequency of occurrence are categorized into the first category;	categorizing a plurality of syntax elements of video content into first and second categories based on the frequency of occurrence threshold, wherein syntax elements which occur greater than the frequency of occurrence threshold are categorized into the second category and syntax elements which occur less than the frequency of occurrence are categorized into the first category. The following specifications provide further evidence of how each of the Accused Products operates:

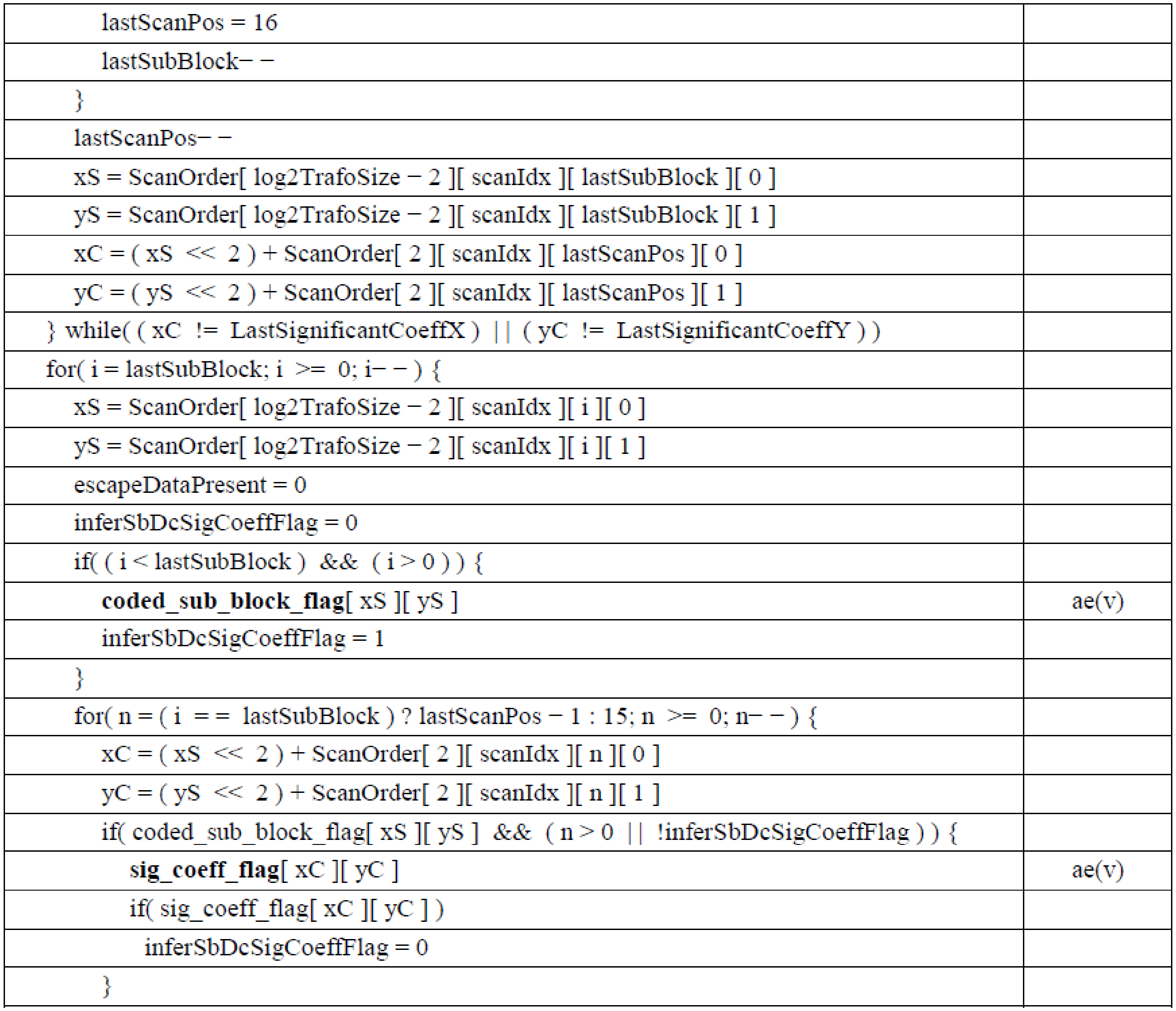
EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

<table>
<tr><th>U.S. PATENT NO. 9,036,701</th><th>ASUS ACCUSED PRODUCTS</th></tr>
<tr><td></td><td>


<b>7.3.8.11 Residual coding syntax</b>

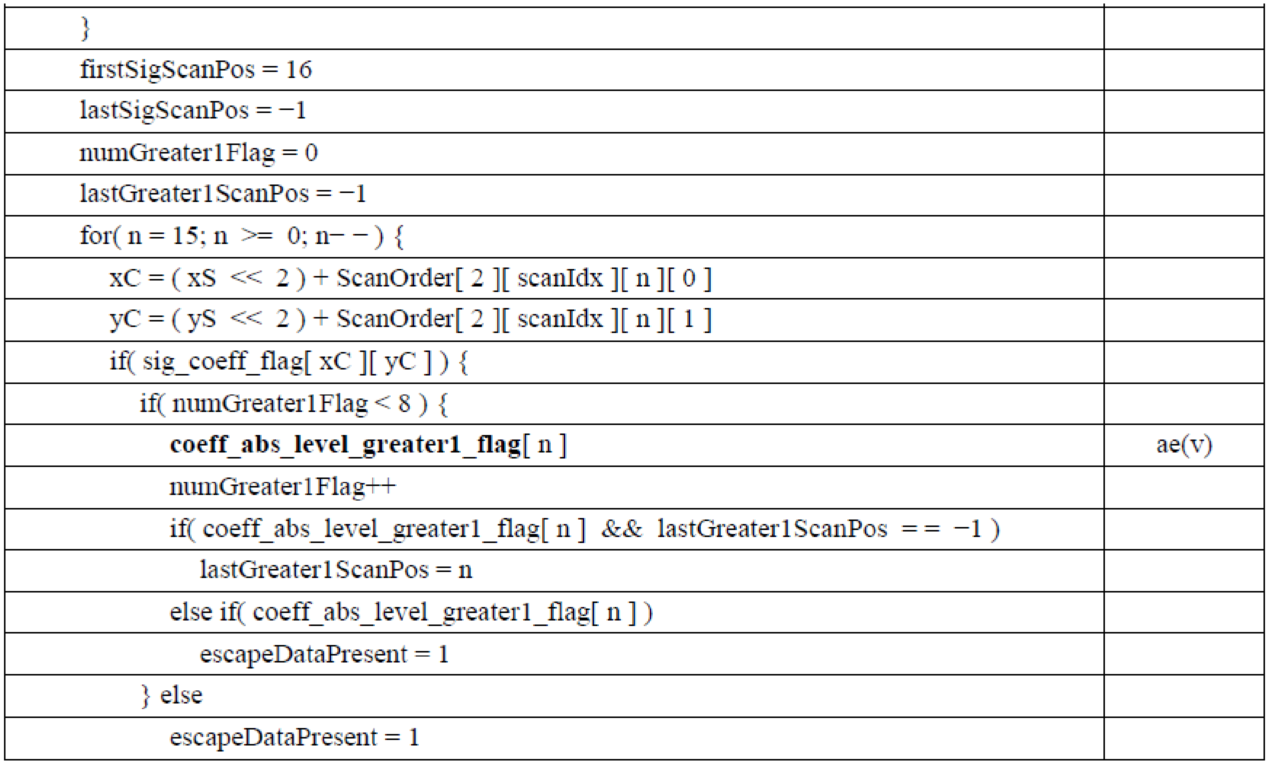
<table>
<tr><td>residual_coding(x0, y0, log2TrafoSize, cIdx) {</td><td><b>Descriptor</b></td></tr>
<tr><td>if(transform_skip_enabled_flag &amp;&amp; !cu_transquant_bypass_flag &amp;&amp; (log2TrafoSize &lt;= Log2MaxTransformSkipSize))</td><td></td></tr>
<tr><td><b>transform_skip_flag</b>[x0][y0][cIdx]</td><td>ae(v)</td></tr>
<tr><td>if(CuPredMode[x0][y0] = = MODE_INTER &amp;&amp; explicit_rdpcm_enabled_flag &amp;&amp; (transform_skip_flag[x0][y0][cIdx] | | cu_transquant_bypass_flag)) {</td><td></td></tr>
<tr><td><b>explicit_rdpcm_flag</b>[x0][y0][cIdx]</td><td>ae(v)</td></tr>
<tr><td>if(explicit_rdpcm_flag[x0][y0][cIdx])</td><td></td></tr>
<tr><td><b>explicit_rdpcm_dir_flag</b>[x0][y0][cIdx]</td><td>ae(v)</td></tr>
<tr><td>}</td><td></td></tr>
<tr><td><b>last_sig_coeff_x_prefix</b></td><td>ae(v)</td></tr>
<tr><td><b>last_sig_coeff_y_prefix</b></td><td>ae(v)</td></tr>
<tr><td>if(last_sig_coeff_x_prefix &gt; 3)</td><td></td></tr>
<tr><td><b>last_sig_coeff_x_suffix</b></td><td>ae(v)</td></tr>
<tr><td>if(last_sig_coeff_y_prefix &gt; 3)</td><td></td></tr>
<tr><td><b>last_sig_coeff_y_suffix</b></td><td>ae(v)</td></tr>
<tr><td>lastScanPos = 16</td><td></td></tr>
<tr><td>lastSubBlock = (1 &lt;&lt; (log2TrafoSize − 2)) * (1 &lt;&lt; (log2TrafoSize − 2)) − 1</td><td></td></tr>
<tr><td>do {</td><td></td></tr>
<tr><td>if(lastScanPos = = 0) {</td><td></td></tr>
</table>
</td></tr>
</table>

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
	

lastScanPos = 16	
lastSubBlock− −	
}	
lastScanPos− −	
xS = ScanOrder[log2TrafoSize − 2][scanIdx][lastSubBlock][0]	
yS = ScanOrder[log2TrafoSize − 2][scanIdx][lastSubBlock][1]	
xC = (xS << 2) + ScanOrder[2][scanIdx][lastScanPos][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][lastScanPos][1]	
} while((xC != LastSignificantCoeffX) || (yC != LastSignificantCoeffY))	
for(i = lastSubBlock; i >= 0; i− −) {	
xS = ScanOrder[log2TrafoSize − 2][scanIdx][i][0]	
yS = ScanOrder[log2TrafoSize − 2][scanIdx][i][1]	
escapeDataPresent = 0	
inferSbDcSigCoeffFlag = 0	
if((i < lastSubBlock) && (i > 0)) {	
coded_sub_block_flag[xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	
}	
for(n = (i = = lastSubBlock) ? lastScanPos − 1 : 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(coded_sub_block_flag[xS][yS] && (n > 0 || !inferSbDcSigCoeffFlag)) {	
sig_coeff_flag[xC][yC]	ae(v)
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	

Exhibit 15
United States Patent No. 9,036,701
Claim Chart for Infringement of Claim 1 by ASUS Accused Products

U.S. Patent No. 9,036,701	ASUS Accused Products
	

}	
firstSigScanPos = 16	
lastSigScanPos = −1	
numGreater1Flag = 0	
lastGreater1ScanPos = −1	
for(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	
if(numGreater1Flag < 8) {	
coeff_abs_level_greater1_flag[n]	ae(v)
numGreater1Flag++	
if(coeff_abs_level_greater1_flag[n] && lastGreater1ScanPos = = −1)	
lastGreater1ScanPos = n	
else if(coeff_abs_level_greater1_flag[n])	
escapeDataPresent = 1	
} else	
escapeDataPresent = 1	

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS

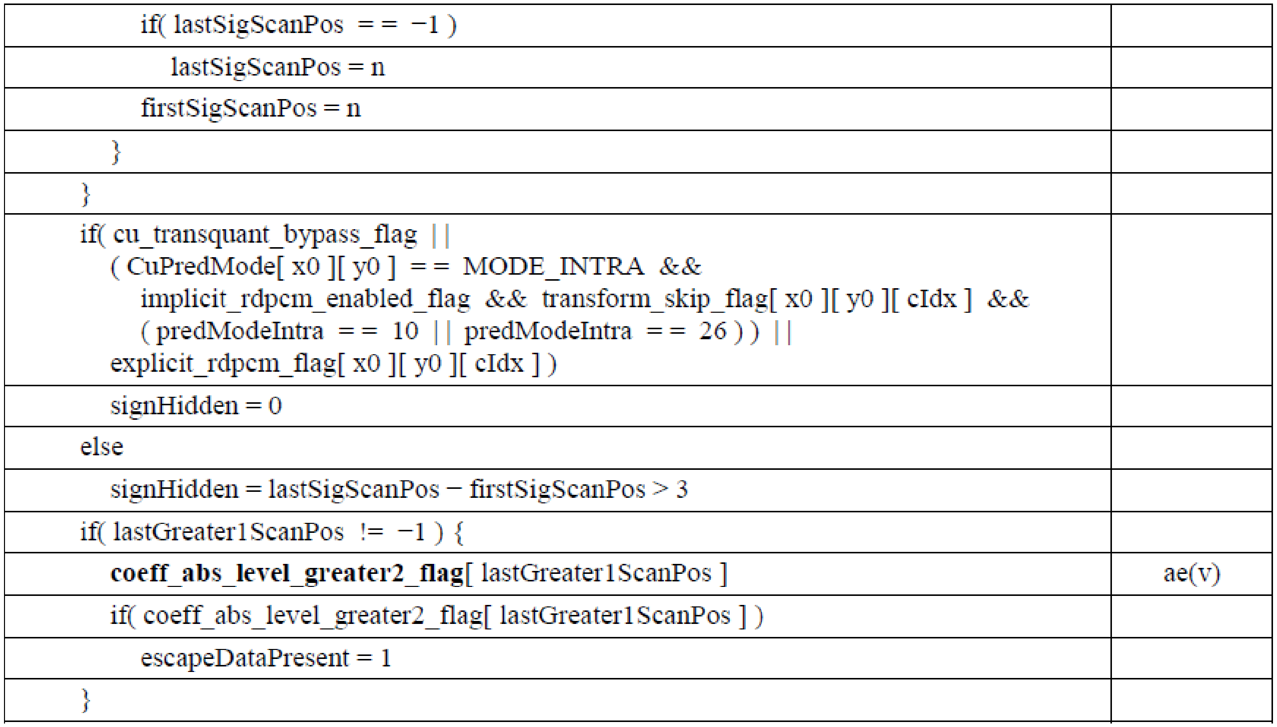
	
	if(lastSigScanPos = = −1)
	lastSigScanPos = n
	firstSigScanPos = n
	}
	}
	if(cu_transquant_bypass_flag || (CuPredMode[x0][y0] = = MODE_INTRA && implicit_rdpcm_enabled_flag && transform_skip_flag[x0][y0][cIdx] && (predModeIntra = = 10 || predModeIntra = = 26)) || explicit_rdpcm_flag[x0][y0][cIdx])
	signHidden = 0
	else
	signHidden = lastSigScanPos − firstSigScanPos > 3
	if(lastGreater1ScanPos != −1) {
	coeff_abs_level_greater2_flag[lastGreater1ScanPos] ae(v)
	if(coeff_abs_level_greater2_flag[lastGreater1ScanPos])
	escapeDataPresent = 1
	}

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

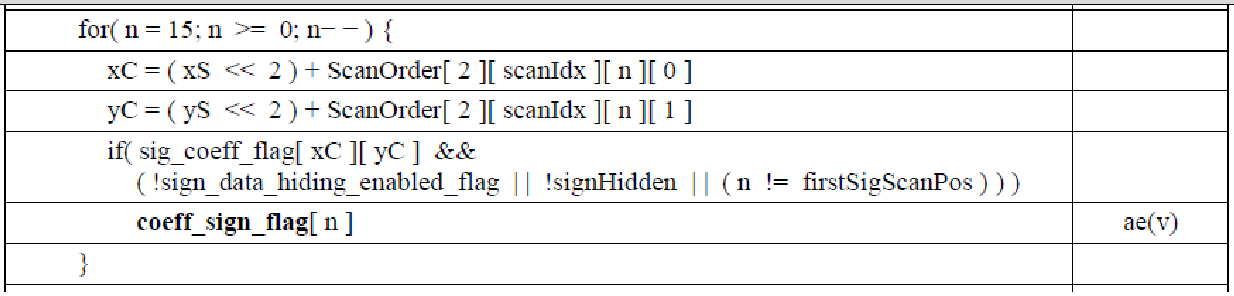
<table>
<tr><th>U.S. PATENT NO. 9,036,701</th><th>ASUS ACCUSED PRODUCTS</th></tr>
<tr><td></td><td>

<table>
<tr><td>for(n = 15; n >= 0; n− −) {</td><td></td></tr>
<tr><td>xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]</td><td></td></tr>
<tr><td>yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]</td><td></td></tr>
<tr><td>if(sig_coeff_flag[xC][yC] &&
(!sign_data_hiding_enabled_flag || !signHidden || (n != firstSigScanPos)))</td><td></td></tr>
<tr><td><b>coeff_sign_flag</b>[n]</td><td>ae(v)</td></tr>
<tr><td>}</td><td></td></tr>
</table>
</td></tr>
</table>

EXHIBIT 15

UNITED STATES PATENT NO. 9,036,701

CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

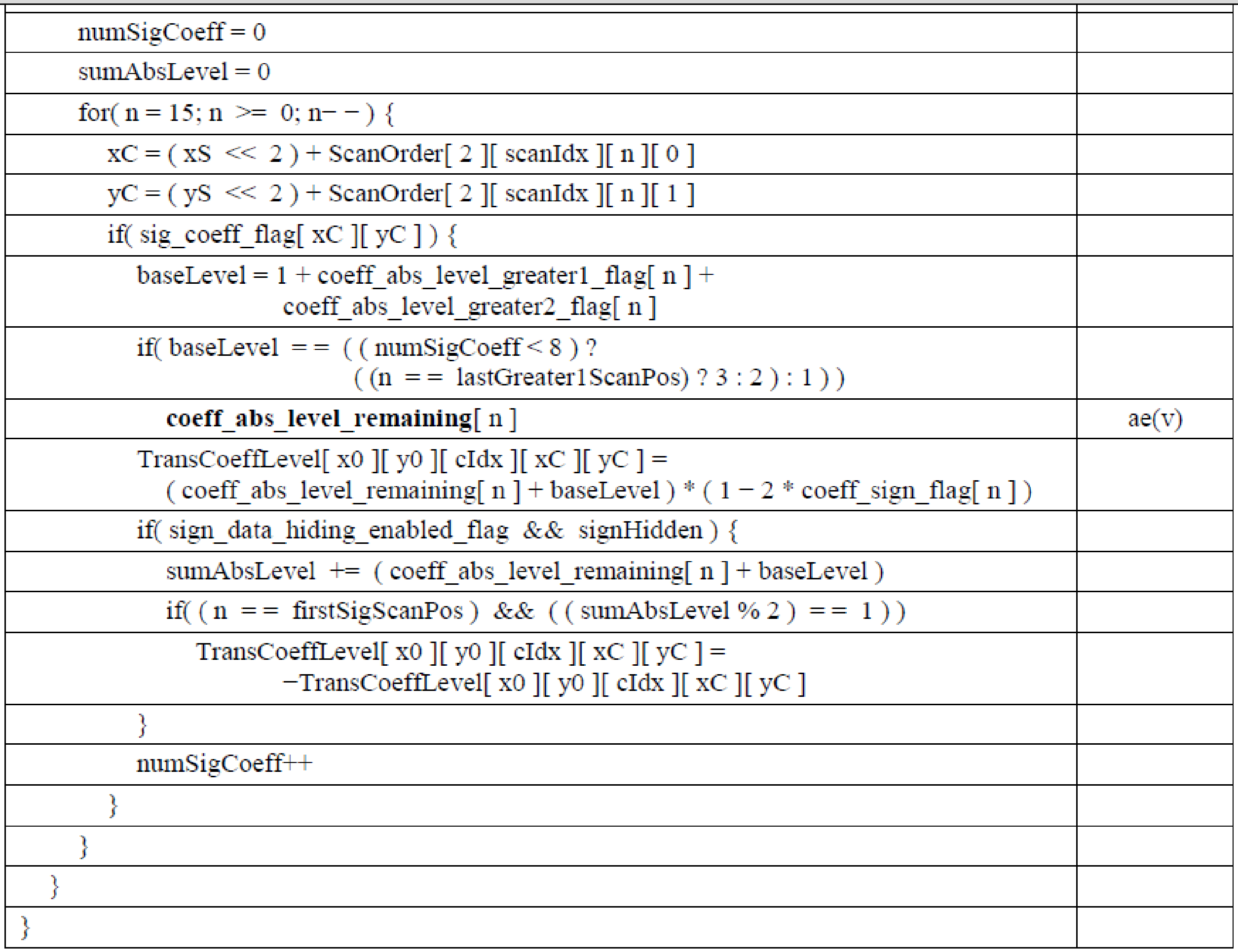
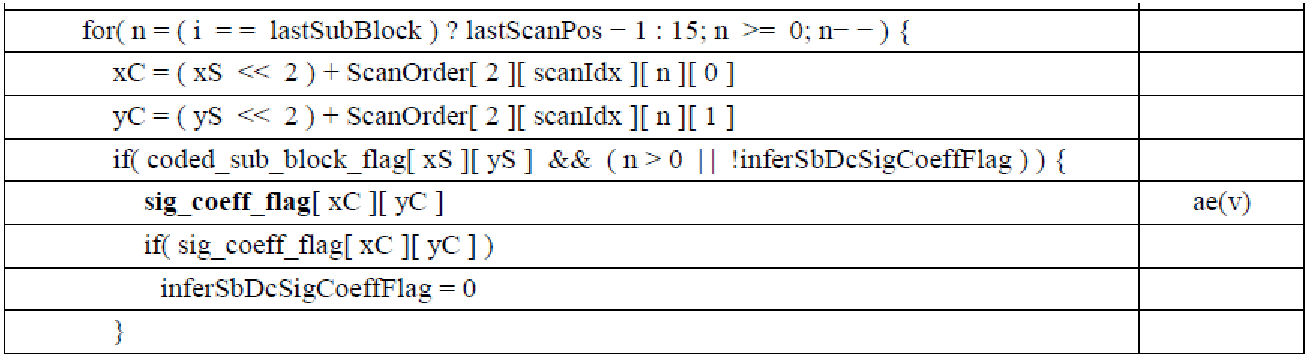
<table>
<tr><th>U.S. PATENT NO. 9,036,701</th><th>ASUS ACCUSED PRODUCTS</th></tr>
<tr><td></td><td>

<table>
<tr><td>numSigCoeff = 0</td><td></td></tr>
<tr><td>sumAbsLevel = 0</td><td></td></tr>
<tr><td>for(n = 15; n >= 0; n− −) {</td><td></td></tr>
<tr><td> xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]</td><td></td></tr>
<tr><td> yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]</td><td></td></tr>
<tr><td> if(sig_coeff_flag[xC][yC]) {</td><td></td></tr>
<tr><td>   baseLevel = 1 + coeff_abs_level_greater1_flag[n] + coeff_abs_level_greater2_flag[n]</td><td></td></tr>
<tr><td>   if(baseLevel = = ((numSigCoeff < 8) ? ((n = = lastGreater1ScanPos) ? 3 : 2) : 1))</td><td></td></tr>
<tr><td>   <b>coeff_abs_level_remaining</b>[n]</td><td>ae(v)</td></tr>
<tr><td>   TransCoeffLevel[x0][y0][cIdx][xC][yC] = (coeff_abs_level_remaining[n] + baseLevel) * (1 − 2 * coeff_sign_flag[n])</td><td></td></tr>
<tr><td>   if(sign_data_hiding_enabled_flag && signHidden) {</td><td></td></tr>
<tr><td>   sumAbsLevel += (coeff_abs_level_remaining[n] + baseLevel)</td><td></td></tr>
<tr><td>   if((n = = firstSigScanPos) && ((sumAbsLevel % 2) = = 1))</td><td></td></tr>
<tr><td>   TransCoeffLevel[x0][y0][cIdx][xC][yC] = −TransCoeffLevel[x0][y0][cIdx][xC][yC]</td><td></td></tr>
<tr><td>   }</td><td></td></tr>
<tr><td>   numSigCoeff++</td><td></td></tr>
<tr><td> }</td><td></td></tr>
<tr><td>}</td><td></td></tr>
<tr><td>}</td><td></td></tr>
<tr><td>}</td><td></td></tr>
</table>
ITU-T Rec. H.265 (12/2016) High efficiency video coding at pp. 58-60.
As set out in the Residual Coding Syntax in section 7.3.8.11, an H.265 compliant decoder parses <i>sig_coeff_flags</i> from the bit stream according to the logic reproduced below.
</td></tr>
</table>

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS



for(n = (i = = lastSubBlock) ? lastScanPos − 1 : 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(coded_sub_block_flag[xS][yS] && (n > 0 || !inferSbDcSigCoeffFlag)) {	
sig_coeff_flag[xC][yC]	ae(v)
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	

ITU-T Rec. H.265 (12/2016) High efficiency video coding at p. 59.

7.4.9.11 Residual coding semantics

. . .

sig_coeff_flag[xC][yC] specifies for the transform coefficient location (xC, yC) within the current transform block whether the corresponding transform coefficient level at the location (xC, yC) is non-zero as follows:

– If sig_coeff_flag[xC][yC] is equal to 0, the transform coefficient level at the location (xC, yC) is set equal to 0.

– Otherwise (sig_coeff_flag[xC][yC] is equal to 1), the transform coefficient level at the location (xC, yC) has a non-zero value.

When sig_coeff_flag[xC][yC] is not present, it is inferred as follows:

– If (xC, yC) is the last significant location (LastSignificantCoeffX, LastSignificantCoeffY) in scan order or all of the following conditions are true, sig_coeff_flag[xC][yC] is inferred to be equal to 1:

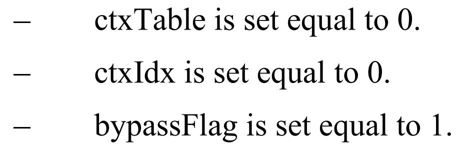
  – (xC & 3, yC & 3) is equal to (0, 0).

  – inferSbDcSigCoeffFlag is equal to 1.

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
	– coded_sub_block_flag[xS][yS] is equal to 1. – Otherwise, sig_coeff_flag[xC][yC] is inferred to be equal to 0. . . . ITU-T Rec. H.265 (12/2016) High efficiency video coding at p. 111. **9.3.4.2 Derivation process for ctxTable, ctxIdx and bypassFlag** **9.3.4.2.1 General** Input to this process is the position of the current bin within the bin string, binIdx. Outputs of this process are ctxTable, ctxIdx and bypassFlag. The values of ctxTable, ctxIdx and bypassFlag are derived as follows based on the entries for binIdx of the corresponding syntax element in Table 9-48: – If the entry in Table 9-48 is not equal to "bypass", "terminate" or "na", the values of binIdx are decoded by invoking the DecodeDecision process as specified in clause 9.3.4.3.2 and the following applies: – ctxTable is specified in Table 9-4. – The variable ctxInc is specified by the corresponding entry in Table 9-48 and when more than one value is listed in Table 9-48 for a binIdx, the assignment process for ctxInc for that binIdx is further specified in the clauses given in parenthesis. – The variable ctxIdxOffset is specified by the lowest value of ctxIdx in Table 9-4 depending on the current value of initType. – ctxIdx is set equal to the sum of ctxInc and ctxIdxOffset. – bypassFlag is set equal to 0. – Otherwise, if the entry in Table 9-48 is equal to "bypass", the values of binIdx are decoded by invoking the DecodeBypass process as specified in clause 9.3.4.3.4 and the following applies:

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

<table>
<tr><th>U.S. PATENT NO. 9,036,701</th><th>ASUS ACCUSED PRODUCTS</th></tr>
<tr><td></td><td>


– ctxTable is set equal to 0.
– ctxIdx is set equal to 0.
– bypassFlag is set equal to 1.

ITU-T Rec. H.265 (12/2016) High efficiency video coding at p. 225.

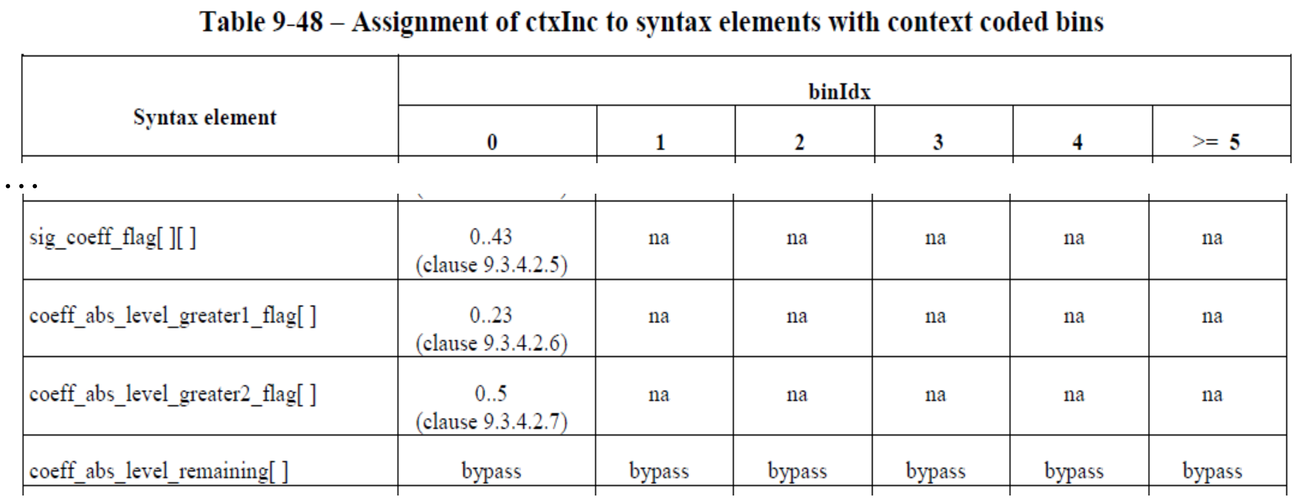


Table 9-48 – Assignment of ctxInc to syntax elements with context coded bins

<table>
<tr><th rowspan="2">Syntax element</th><th colspan="6">binIdx</th></tr>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>&gt;= 5</th></tr>
<tr><td>. . .</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>sig_coeff_flag[ ][ ]</td><td>0..43
(clause 9.3.4.2.5)</td><td>na</td><td>na</td><td>na</td><td>na</td><td>na</td></tr>
<tr><td>coeff_abs_level_greater1_flag[ ]</td><td>0..23
(clause 9.3.4.2.6)</td><td>na</td><td>na</td><td>na</td><td>na</td><td>na</td></tr>
<tr><td>coeff_abs_level_greater2_flag[ ]</td><td>0..5
(clause 9.3.4.2.7)</td><td>na</td><td>na</td><td>na</td><td>na</td><td>na</td></tr>
<tr><td>coeff_abs_level_remaining[ ]</td><td>bypass</td><td>bypass</td><td>bypass</td><td>bypass</td><td>bypass</td><td>bypass</td></tr>
</table>

ITU-T Rec. H.265 (12/2016) High efficiency video coding at pp. 225-227.
</td></tr>
<tr><td><b>[c]</b> entropy coding symbols that correspond to the first category of syntax elements and that have been subjected to a context update; and</td><td>Each of the Accused Products, such as the ASUS Q543MV, performs a method comprising entropy coding symbols that correspond to the first category of syntax elements and that have been subjected to a context update.

For example, and without limitation, the H.265 Standard specifies the following regarding the decoding process. Each of the Accused Products performs a method comprising entropy coding symbols that correspond to the first category of syntax elements and that have been subjected to a context update.</td></tr>
</table>

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
	The following specifications provide further evidence of how each of the Accused Products operates: **9.3.4.2 Derivation process for ctxTable, ctxIdx and bypassFlag** **9.3.4.2.1 General** Input to this process is the position of the current bin within the bin string, binIdx. Outputs of this process are ctxTable, ctxIdx and bypassFlag. The values of ctxTable, ctxIdx and bypassFlag are derived as follows based on the entries for binIdx of the corresponding syntax element in Table 9-48: – If the entry in Table 9-48 is not equal to "bypass", "terminate" or "na", the values of binIdx are decoded by invoking the DecodeDecision process as specified in clause 9.3.4.3.2 and the following applies: – ctxTable is specified in Table 9-4. – The variable ctxInc is specified by the corresponding entry in Table 9-48 and when more than one value is listed in Table 9-48 for a binIdx, the assignment process for ctxInc for that binIdx is further specified in the clauses given in parenthesis. – The variable ctxIdxOffset is specified by the lowest value of ctxIdx in Table 9-4 depending on the current value of initType. – ctxIdx is set equal to the sum of ctxInc and ctxIdxOffset. – bypassFlag is set equal to 0. . . . ITU-T Rec. H.265 (12/2016) High efficiency video coding at p. 225. **9.3.4.3 Arithmetic decoding process** **9.3.4.3.1 General**

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
	Inputs to this process are ctxTable, ctxIdx and bypassFlag, as derived in clause 9.3.4.2, and the state variables ivlCurrRange and ivlOffset of the arithmetic decoding engine. Output of this process is the value of the bin. Figure 9-5 illustrates the whole arithmetic decoding process for a single bin. For decoding the value of a bin, the context index table ctxTable and the ctxIdx are passed to the arithmetic decoding process DecodeBin(ctxTable, ctxIdx), which is specified as follows: – If bypassFlag is equal to 1, DecodeBypass( ) as specified in clause 9.3.4.3.4 is invoked. – Otherwise, if bypassFlag is equal to 0, ctxTable is equal to 0 and ctxIdx is equal to 0, DecodeTerminate( ) as specified in clause 9.3.4.3.5 is invoked. – Otherwise (bypassFlag is equal to 0 and ctxTable is not equal to 0), DecodeDecision( ) as specified in clause 9.3.4.3.2 is invoked. . . . ITU-T Rec. H.265 (12/2016) High efficiency video coding at p. 232. **9.3.4.2.6 Derivation process of ctxInc for the syntax element coeff_abs_level_greater1_flag** Inputs to this process are the colour component index cIdx, the current sub-block scan index i and the current coefficient scan index n within the current sub-block. Output of this process is the variable ctxInc. . . . ITU-T Rec. H.265 (12/2016) High efficiency video coding at p 231. **9.3.4.2.7 Derivation process of ctxInc for the syntax element coeff_abs_level_greater2_flag** Inputs to this process are the colour component index cIdx, the current sub-block scan index i and the current coefficient scan index n within the current sub-block.

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS

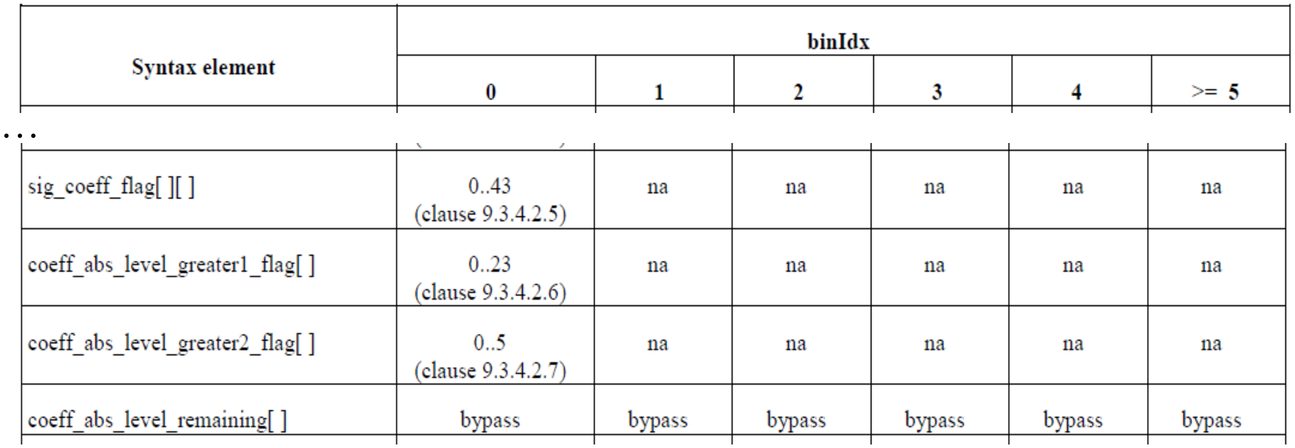
Output of this process is the variable ctxInc.

. . .

ITU-T Rec. H.265 (12/2016) High efficiency video coding at p. 232.

The symbols that correspond to the first category of syntax elements and that have been subjected to a context update are the **coeff_abs_level_greater1_flag**, and **coeff_abs_level_greater2_flag** that are used to represent, at least in part, the *non-zero transform coefficient levels* in the first category, as shown in Table 9-48 and described in section 9.3.2.5 of the H.265 Standard reproduced below, in which the process for calculating the context for each of **coeff_abs_level_greater1_flag** and **coeff_abs_level_greater2_flag** is provided.

Table 9-48 – Assignment of ctxInc to syntax elements with context coded bins



Syntax element	binIdx					
	0	1	2	3	4	>= 5
. . .						
sig_coeff_flag[ ][ ]	0..43 (clause 9.3.4.2.5)	na	na	na	na	na
coeff_abs_level_greater1_flag[ ]	0..23 (clause 9.3.4.2.6)	na	na	na	na	na
coeff_abs_level_greater2_flag[ ]	0..5 (clause 9.3.4.2.7)	na	na	na	na	na
coeff_abs_level_remaining[ ]	bypass	bypass	bypass	bypass	bypass	bypass

ITU-T Rec. H.265 (12/2016) High efficiency video coding at pp. 225-227.

9.3.2.5 Synchronization process for context variables, Rice parameter initialization states, and palette predictor variables

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
	Inputs to this process are: – The variables tableStateSync and tableMPSSync containing the values of the variables pStateIdx and valMps used in the storage process of context variables that are assigned to all syntax elements in clauses 7.3.8.1 through 7.3.8.12, except end_of_slice_segment_flag, end_of_subset_one_bit and pcm_flag. – The variable tableStatCoeffSync containing the values of the variables StatCoeff[k] used in the storage process of context variables and Rice parameter initialization states. – The variables PredictorPaletteSizeSync and tablePredictorPaletteEntriesSync containing the values used in the storage process of palette predictor variables. Outputs of this process are: – The initialized CABAC context variables indexed by ctxTable and ctxIdx. – The initialized Rice parameter initialization states StatCoeff indexed by k. – The palette predictor variables, PredictorPaletteSize and PredictorPaletteEntries. . . . ITU-T Rec. H.265 (12/2016) High efficiency video coding at p. 215.
[d] entropy coding symbols that correspond to the second category of syntax elements and that have bypassed context updating.	Each of the Accused Products, such as the ASUS Q543MV, performs a method comprising entropy coding symbols that correspond to the second category of syntax elements and that have bypassed context updating. For example, and without limitation, the H.265 Standard specifies the following regarding the decoding process. Each of the Accused Products performs a method comprising entropy coding symbols that correspond to the second category of syntax elements and that have bypassed context updating. The following specifications provide further evidence of how each of the Accused Products operates: **9.3.4.2 Derivation process for ctxTable, ctxIdx and bypassFlag**

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS

9.3.4.2.1 General

Input to this process is the position of the current bin within the bin string, binIdx.

Outputs of this process are ctxTable, ctxIdx and bypassFlag.

The values of ctxTable, ctxIdx and bypassFlag are derived as follows based on the entries for binIdx of the corresponding syntax element in Table 9-48:

. . .

– Otherwise, if the entry in Table 9-48 is equal to "bypass", the values of binIdx are decoded by invoking the DecodeBypass process as specified in clause 9.3.4.3.4 and the following applies:

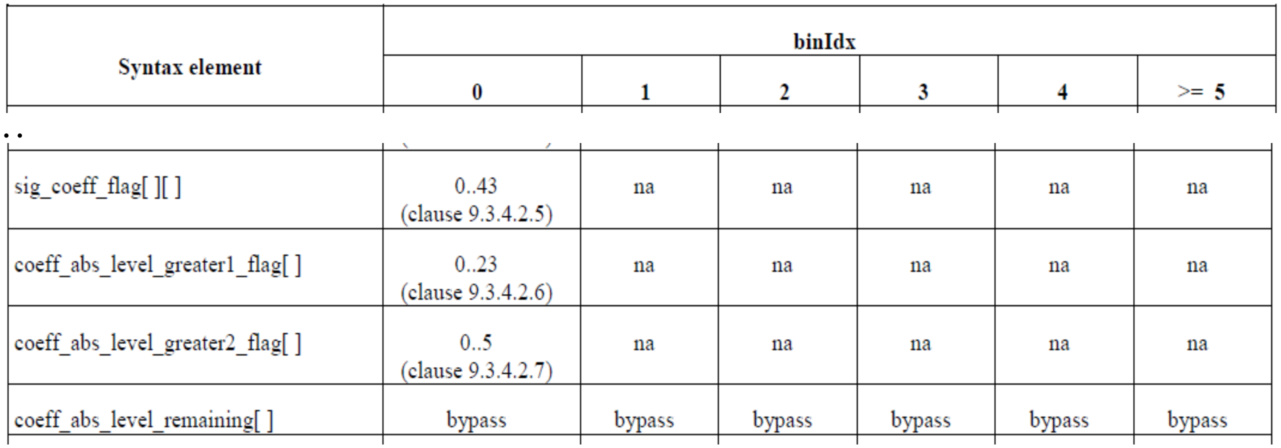
  – ctxTable is set equal to 0.

  – ctxIdx is set equal to 0.

  – bypassFlag is set equal to 1.

. . .

Table 9-48 – Assignment of ctxInc to syntax elements with context coded bins



Syntax element	binIdx					
	0	1	2	3	4	>= 5
. . .						
sig_coeff_flag[ ][ ]	0..43 (clause 9.3.4.2.5)	na	na	na	na	na
coeff_abs_level_greater1_flag[ ]	0..23 (clause 9.3.4.2.6)	na	na	na	na	na
coeff_abs_level_greater2_flag[ ]	0..5 (clause 9.3.4.2.7)	na	na	na	na	na
coeff_abs_level_remaining[ ]	bypass	bypass	bypass	bypass	bypass	bypass

ITU-T Rec. H.265 (12/2016) High efficiency video coding at pp. 225-227.

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
	9.3.4.3 Arithmetic decoding process **9.3.4.3.1 General** Inputs to this process are ctxTable, ctxIdx and bypassFlag, as derived in clause 9.3.4.2, and the state variables ivlCurrRange and ivlOffset of the arithmetic decoding engine. Output of this process is the value of the bin. Figure 9-5 illustrates the whole arithmetic decoding process for a single bin. For decoding the value of a bin, the context index table ctxTable and the ctxIdx are passed to the arithmetic decoding process DecodeBin(ctxTable, ctxIdx), which is specified as follows: – If bypassFlag is equal to 1, DecodeBypass( ) as specified in clause 9.3.4.3.4 is invoked. – Otherwise, if bypassFlag is equal to 0, ctxTable is equal to 0 and ctxIdx is equal to 0, DecodeTerminate( ) as specified in clause 9.3.4.3.5 is invoked. – Otherwise (bypassFlag is equal to 0 and ctxTable is not equal to 0), DecodeDecision( ) as specified in clause 9.3.4.3.2 is invoked. . . . ITU-T Rec. H.265 (12/2016) High efficiency video coding at p. 232.

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

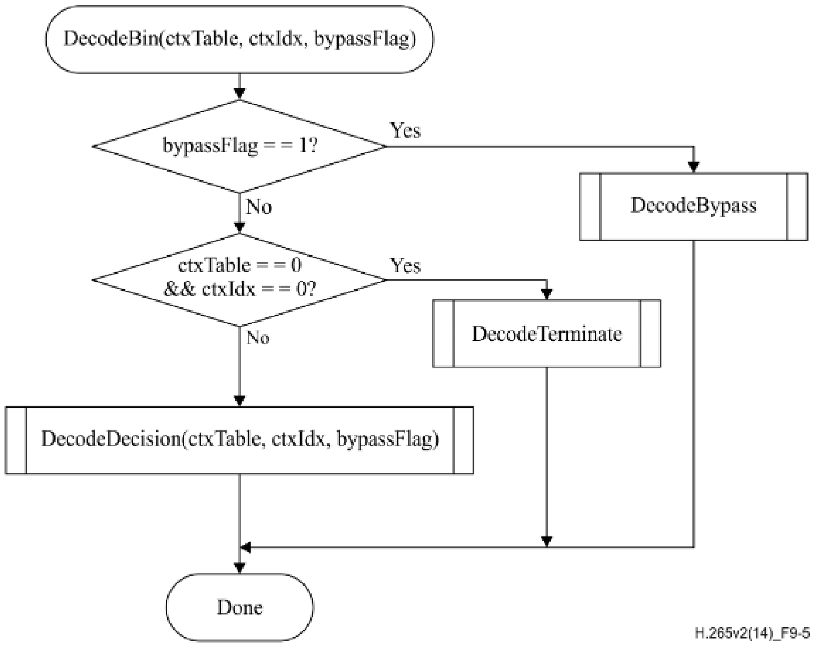
U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
	 DecodeBin(ctxTable, ctxIdx, bypassFlag) bypassFlag = = 1? Yes DecodeBypass No ctxTable = = 0 && ctxIdx = = 0? Yes DecodeTerminate No DecodeDecision(ctxTable, ctxIdx, bypassFlag) Done H.265v2(14)_F9-5 **Figure 9-5 – Overview of the arithmetic decoding process for a single bin (informative)** NOTE – Arithmetic coding is based on the principle of recursive interval subdivision. Given a probability estimation p(0) and p(1) = 1 − p(0) of a binary decision (0, 1), an initially given code sub-interval with the range ivlCurrRange will be subdivided into two sub-intervals having range p(0) * ivlCurrRange and ivlCurrRange − p(0) * ivlCurrRange, respectively. Depending on the decision, which has been observed, the corresponding sub-interval will be chosen as the new code interval, and a binary code string pointing into that interval will represent the sequence of observed binary decisions. It is useful to distinguish between the most probable symbol (MPS) and the least probable symbol (LPS), so that binary decisions have to be identified as either MPS or LPS, rather than 0 or 1. Given this terminology, each context is specified by the probability pLPS of the LPS and the value of MPS (valMps), which is either 0 or 1. The arithmetic core engine in this Specification has three distinct properties:

EXHIBIT 15
UNITED STATES PATENT NO. 9,036,701
CLAIM CHART FOR INFRINGEMENT OF CLAIM 1 BY ASUS ACCUSED PRODUCTS

U.S. PATENT NO. 9,036,701	ASUS ACCUSED PRODUCTS
	– The probability estimation is performed by means of a finite-state machine with a table-based transition process between 64 different representative probability states { pLPS(pStateIdx) | 0 <= pStateIdx < 64 } for the LPS probability pLPS. The numbering of the states is arranged in such a way that the probability state with index pStateIdx = 0 corresponds to an LPS probability value of 0.5, with decreasing LPS probability towards higher state indices. – The range ivlCurrRange representing the state of the coding engine is quantized to a small set {Q1,...,Q4} of pre-set quantization values prior to the calculation of the new interval range. Storing a table containing all 64x4 pre-computed product values of Qi * pLPS(pStateIdx) allows a multiplication-free approximation of the product ivlCurrRange * pLPS(pStateIdx). – For syntax elements or parts thereof for which an approximately uniform probability distribution is assumed to be given a separate simplified encoding and decoding bypass process is used. ITU-T Rec. H.265 (12/2016) High efficiency video coding at p. 233.